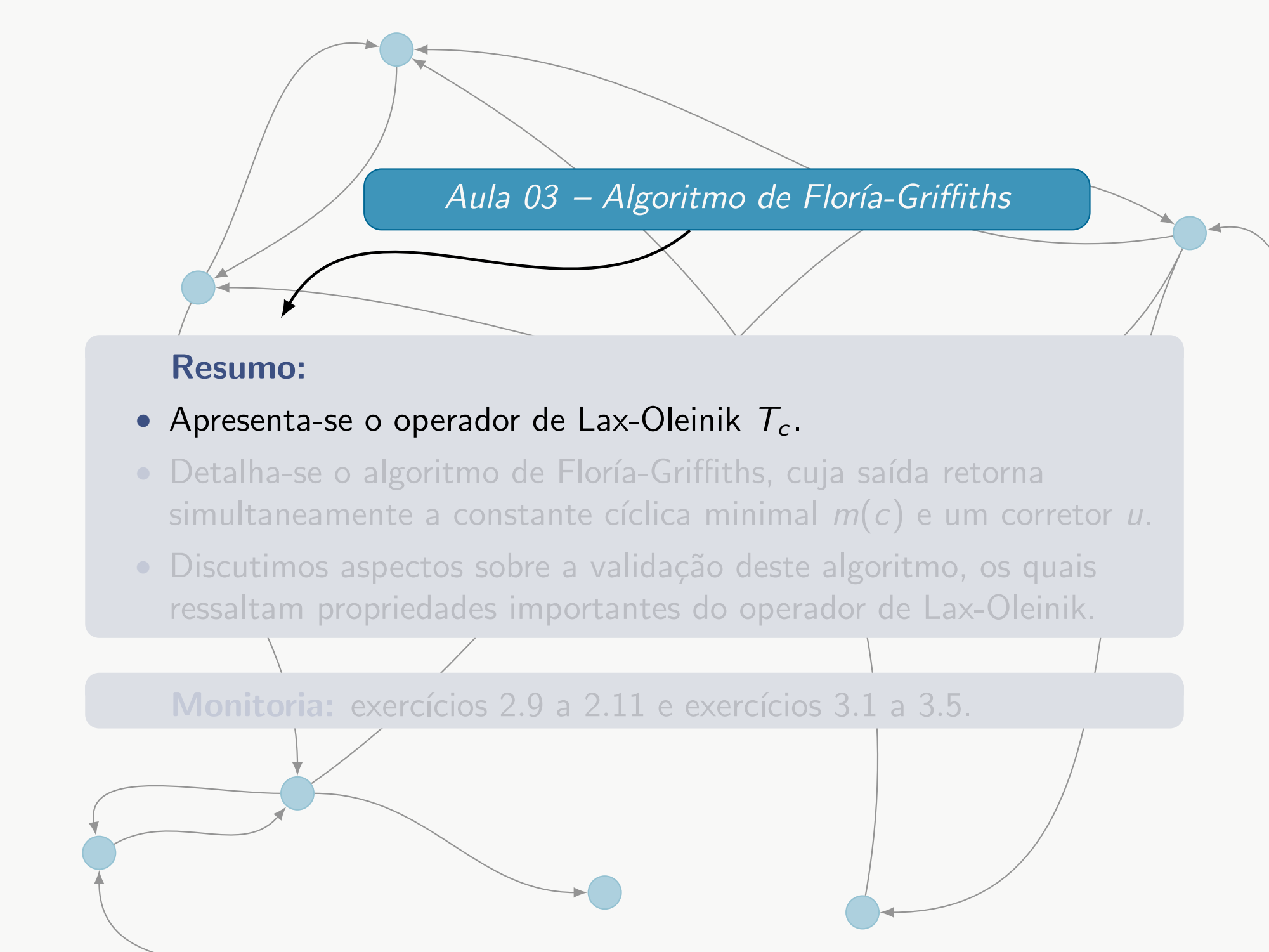
29º Colóquio Brasileiro de Matemática
Rio de Janeiro – Julho, 2013

Otimização de Médias sobre Grafos Orientados

Aula 03 – Algoritmo de Floría-Griffiths

Eduardo Garibaldi João Tiago Assunção Gomes
IMECC, Universidade Estadual de Campinas

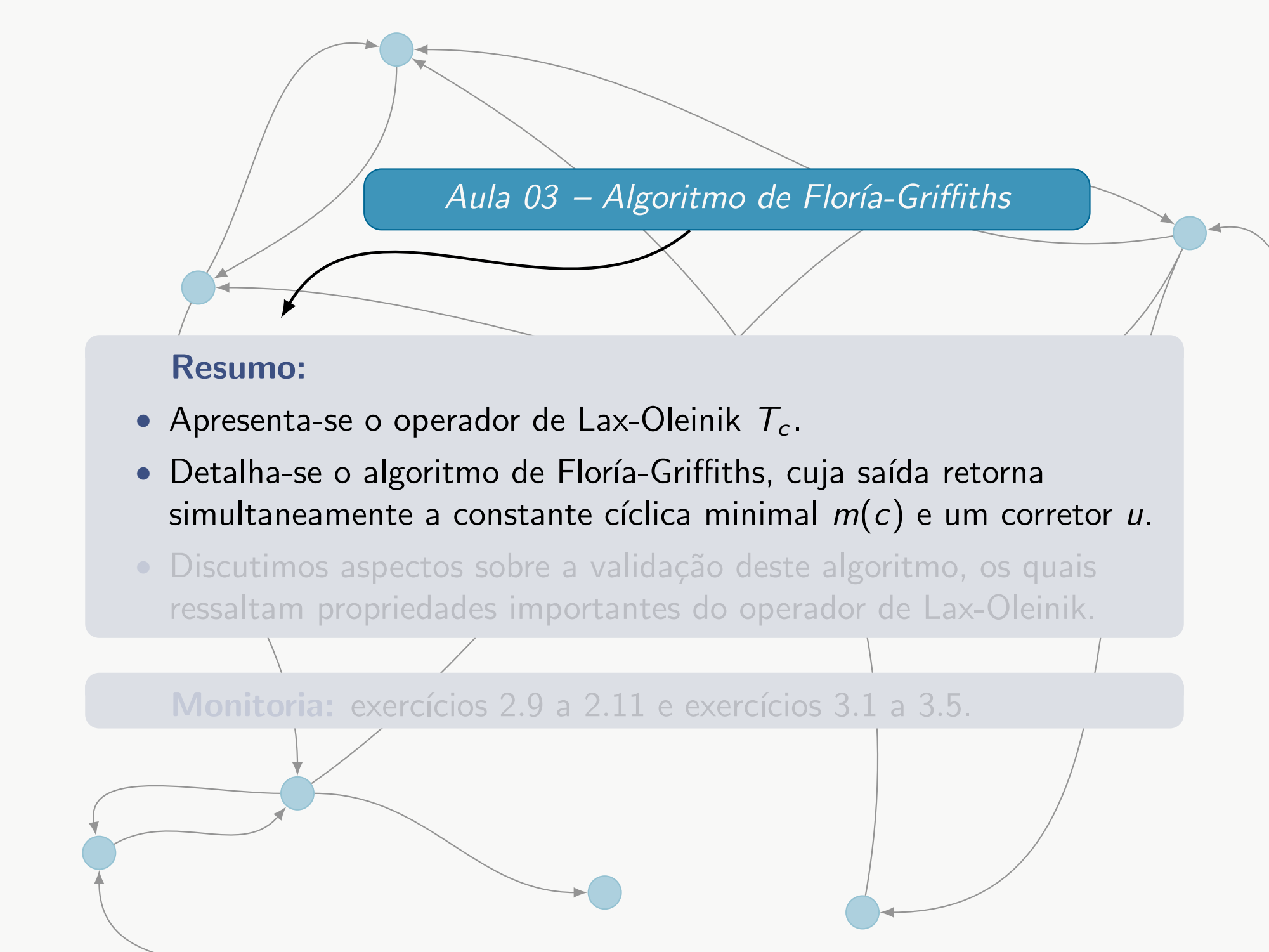


Aula 03 – Algoritmo de Floría-Griffiths

Resumo:

- Apresenta-se o operador de Lax-Oleinik T_c.
- Detalha-se o algoritmo de Floría-Griffiths, cuja saída retorna simultaneamente a constante cíclica minimal $m(c)$ e um corretor u.
- Discutimos aspectos sobre a validação deste algoritmo, os quais ressaltam propriedades importantes do operador de Lax-Oleinik.

Monitoria: exercícios 2.9 a 2.11 e exercícios 3.1 a 3.5.



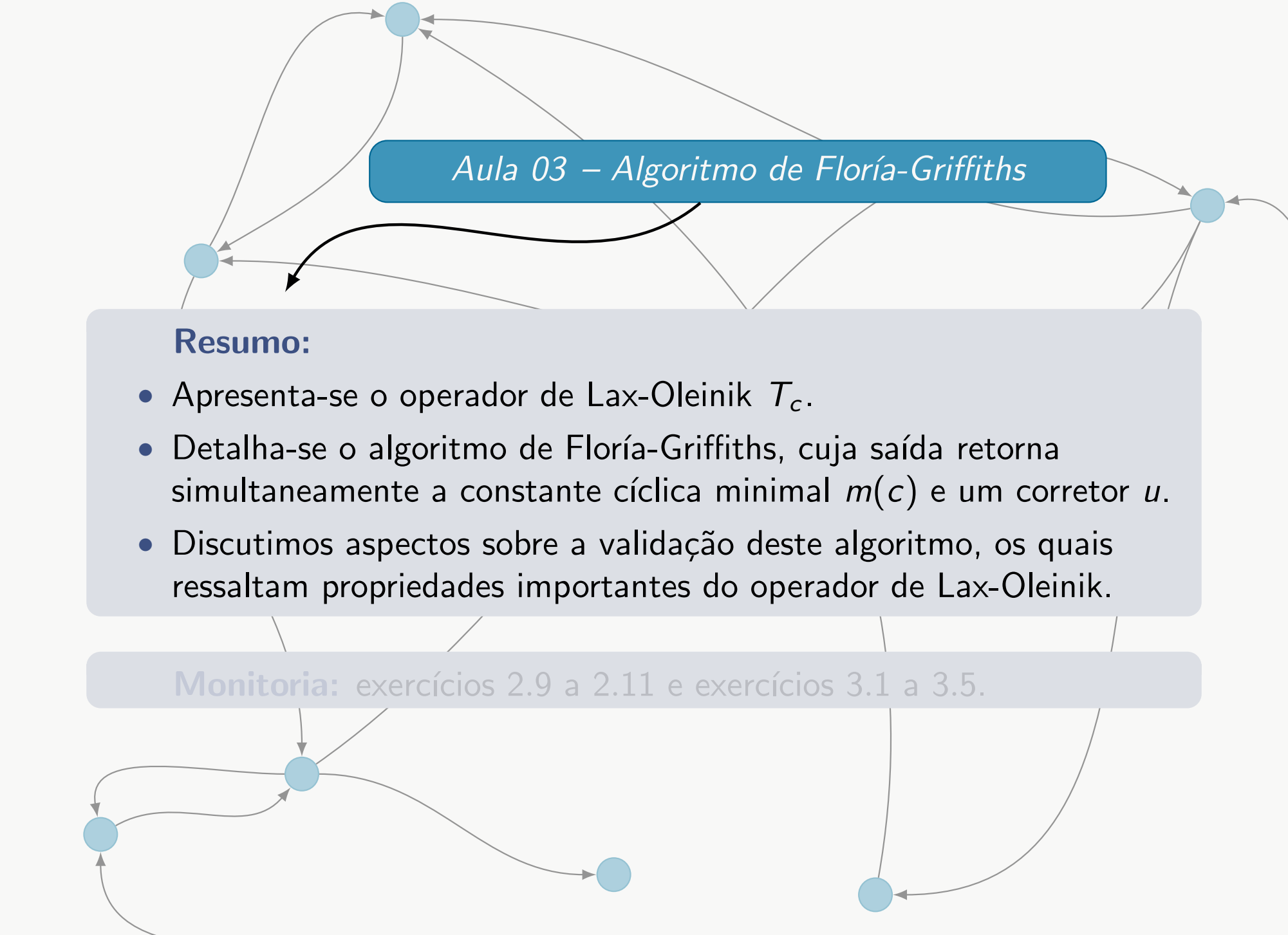
Aula 03 – Algoritmo de Floría-Griffiths

Resumo:

- Apresenta-se o operador de Lax-Oleinik T_c.
- Detalha-se o algoritmo de Floría-Griffiths, cuja saída retorna simultaneamente a constante cíclica minimal $m(c)$ e um corretor u.
- Discutimos aspectos sobre a validação deste algoritmo, os quais ressaltam propriedades importantes do operador de Lax-Oleinik.

Monitoria: exercícios 2.9 a 2.11 e exercícios 3.1 a 3.5.

Aula 03 – Algoritmo de Floría-Griffiths

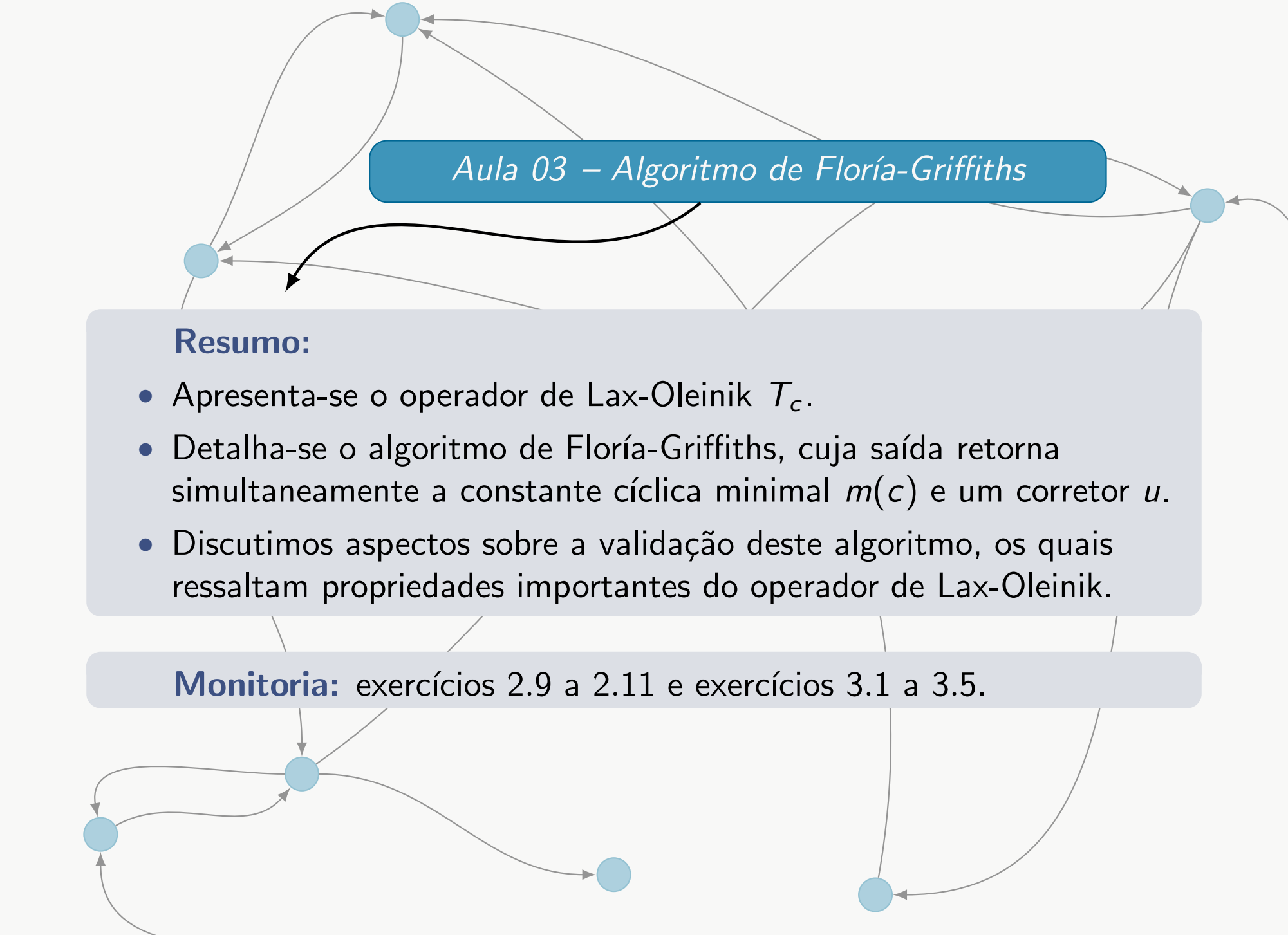


Resumo:

- Apresenta-se o operador de Lax-Oleinik T_c.
- Detalha-se o algoritmo de Floría-Griffiths, cuja saída retorna simultaneamente a constante cíclica minimal $m(c)$ e um corretor u.
- Discutimos aspectos sobre a validação deste algoritmo, os quais ressaltam propriedades importantes do operador de Lax-Oleinik.

Monitoria: exercícios 2.9 a 2.11 e exercícios 3.1 a 3.5.

Aula 03 – Algoritmo de Floría-Griffiths

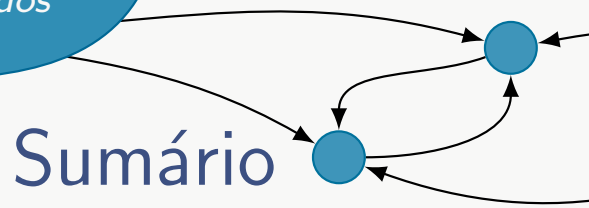


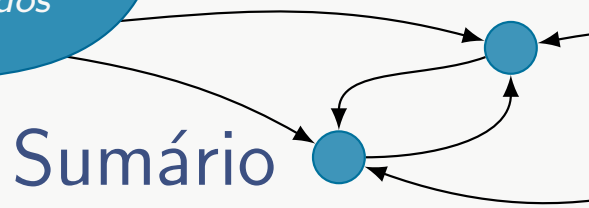
Resumo:

- Apresenta-se o operador de Lax-Oleinik T_c.
- Detalha-se o algoritmo de Floría-Griffiths, cuja saída retorna simultaneamente a constante cíclica minimal $m(c)$ e um corretor u.
- Discutimos aspectos sobre a validação deste algoritmo, os quais ressaltam propriedades importantes do operador de Lax-Oleinik.

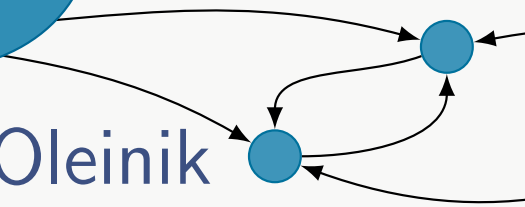
Monitoria: exercícios 2.9 a 2.11 e exercícios 3.1 a 3.5.

Sumário





Sumário



Operador de Lax-Oleinik

Relembrando a definição equivalente de corretor:

$$u(j) \leq \min_{i \overset{G}{\to} j} [u(i) + c(i,j) - m(c)], \qquad \forall\, j \in V(G),$$

Baseando-se nesta reescritura, introduzimos seguinte operador.

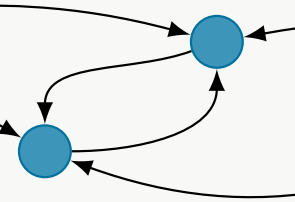
Definição

Seja $c : A(G) \to \mathbb{R}$ função custo arbitrária. O operador de Lax-Oleinik associado T_c age sobre as funções $f : V(G) \to \mathbb{R}$ por

$$\begin{aligned} T_c f(j) &:= \min\{f(i) + c(i,j) \ : \ i \in V(G), M(i,j) = 1\} \\ &= \min\{f(i) + c(i,j) \ : \ (i,j) \in A(G)\} \\ &= \min_{i \overset{G}{\to} j}[f(i) + c(i,j)], \qquad \forall\, j \in V(G). \end{aligned}$$

Observação

Note que u é corretor para um custo c se, e só se, $u \leq T_{c-m(c)}u$.



Operador de Lax-Oleinik

Relembrando a definição equivalente de corretor:

$$u(j) \leq \min_{i \overset{G}{\to} j} [u(i) + c(i,j) - m(c)], \qquad \forall\, j \in V(G),$$

Baseando-se nesta reescritura, introduzimos seguinte operador.

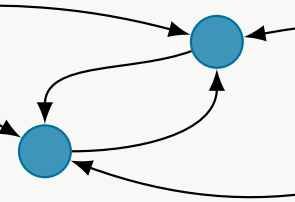
Definição

Seja $c : A(G) \to \mathbb{R}$ função custo arbitrária. O operador de Lax-Oleinik associado T_c age sobre as funções $f : V(G) \to \mathbb{R}$ por

$$\begin{aligned} T_c f(j) &:= \min\{f(i) + c(i,j) \ : \ i \in V(G), M(i,j) = 1\} \\ &= \min\{f(i) + c(i,j) \ : \ (i,j) \in A(G)\} \\ &= \min_{i \overset{G}{\to} j}[f(i) + c(i,j)], \qquad \forall\, j \in V(G). \end{aligned}$$

Observação

Note que u é corretor para um custo c se, e só se, $u \leq T_{c-m(c)} u$.



Operador de Lax-Oleinik

Relembrando a definição equivalente de corretor:

$$u(j) \leq \min_{i \overset{G}{\to} j} [u(i) + c(i,j) - m(c)], \qquad \forall\, j \in V(G),$$

Baseando-se nesta reescritura, introduzimos seguinte operador.

Definição

Seja $c : A(G) \to \mathbb{R}$ função custo arbitrária. O operador de Lax-Oleinik associado T_c age sobre as funções $f : V(G) \to \mathbb{R}$ por

$$\begin{aligned} T_c f(j) &:= \min\{f(i) + c(i,j) \ : \ i \in V(G), M(i,j) = 1\} \\ &= \min\{f(i) + c(i,j) \ : \ (i,j) \in A(G)\} \\ &= \min_{i \overset{G}{\to} j}[f(i) + c(i,j)], \qquad \forall\, j \in V(G). \end{aligned}$$

Observação

Note que u é corretor para um custo c se, e só se, $u \leq T_{c-m(c)} u$.

As propriedades do operador de Lax-Oleinik que queremos aqui destacar estão reunidas abaixo.

Proposição

O operador de Lax-Oleinik verifica:

1. $f \leq g \Rightarrow T_c f \leq T_c g$;
2. $T_c(f \wedge g) = T_c f \wedge T_c g$;
3. $T_{c_k} T_{c_{k-1}} \cdots T_{c_1} T_{c_0} f(j) =$

$$= \min \left\{ f(i_0) + \sum_{r=0}^{k} c_r(i_r, i_{r+1}) : \begin{array}{c} i_0 \xrightarrow{G} \ldots \xrightarrow{G} i_{k+1} = j \\ \text{caminho em } G \\ \text{com ponto final } j \end{array} \right\}.$$

4. *se* $\#V(G) = n$, *então*

$$T_{c-m(c)} f \wedge T^2_{c-m(c)} f \wedge \ldots \wedge T^n_{c-m(c)} f \leq T^k_{c-m(c)} f, \quad \forall\, k \geq n+1.$$

As propriedades do operador de Lax-Oleinik que queremos aqui destacar estão reunidas abaixo.

Proposição

O operador de Lax-Oleinik verifica:

1. $f \leq g \Rightarrow T_c f \leq T_c g$;
2. $T_c(f \wedge g) = T_c f \wedge T_c g$;
3. $T_{c_k} T_{c_{k-1}} \cdots T_{c_1} T_{c_0} f(j) =$

$$= \min \left\{ f(i_0) + \sum_{r=0}^{k} c_r(i_r, i_{r+1}) : \begin{array}{c} i_0 \xrightarrow{G} \ldots \xrightarrow{G} i_{k+1} = j \\ \text{caminho em } G \\ \text{com ponto final } j \end{array} \right\}.$$

4. *se* $\#V(G) = n$, *então*

$$T_{c-m(c)} f \wedge T^2_{c-m(c)} f \wedge \ldots \wedge T^n_{c-m(c)} f \leq T^k_{c-m(c)} f, \quad \forall\, k \geq n+1.$$

As propriedades do operador de Lax-Oleinik que queremos aqui destacar estão reunidas abaixo.

Proposição

O operador de Lax-Oleinik verifica:

1. $f \leq g \Rightarrow T_c f \leq T_c g$;
2. $T_c(f \wedge g) = T_c f \wedge T_c g$;
3. $T_{c_k} T_{c_{k-1}} \cdots T_{c_1} T_{c_0} f(j) =$

$$= \min \left\{ f(i_0) + \sum_{r=0}^{k} c_r(i_r, i_{r+1}) : \begin{array}{c} i_0 \xrightarrow{G} \ldots \xrightarrow{G} i_{k+1} = j \\ \textit{caminho em } G \\ \textit{com ponto final } j \end{array} \right\}.$$

4. *se* $\#V(G) = n$, *então*

$$T_{c-m(c)} f \wedge T^2_{c-m(c)} f \wedge \ldots \wedge T^n_{c-m(c)} f \leq T^k_{c-m(c)} f, \quad \forall\, k \geq n+1.$$

As propriedades do operador de Lax-Oleinik que queremos aqui destacar estão reunidas abaixo.

Proposição

O operador de Lax-Oleinik verifica:

1. $f \leq g \Rightarrow T_c f \leq T_c g$;
2. $T_c(f \wedge g) = T_c f \wedge T_c g$;
3. $T_{c_k} T_{c_{k-1}} \cdots T_{c_1} T_{c_0} f(j) =$

$$= \min \left\{ f(i_0) + \sum_{r=0}^{k} c_r(i_r, i_{r+1}) : \begin{array}{c} i_0 \xrightarrow{G} \ldots \xrightarrow{G} i_{k+1} = j \\ \textit{caminho em } G \\ \textit{com ponto final } j \end{array} \right\}.$$

4. *se* $\#V(G) = n$, *então*

$$T_{c-m(c)} f \wedge T^2_{c-m(c)} f \wedge \ldots \wedge T^n_{c-m(c)} f \leq T^k_{c-m(c)} f, \quad \forall\, k \geq n+1.$$

Observação

Um caso particular do item 3 é

$$T_{c-m_k} T_{c-m_{k-1}} \cdots T_{c-m_1} T_{c-m_0} f(j) =$$

$$= \min_P \left[f(i_P) + (k+1)c(P) - \sum_{r=0}^{k} m_r \right],$$

onde o mínimo é tomado sobre todos os caminhos P em G de comprimento $k+1$ conectando vértice arbitrário i_P ao vértice fixo j.

Ademais, caso $m_k = \ldots = m_1 = m_0 = m(c)$, temos

$$T^{k+1}_{c-m(c)} f(j) = \min_P \left[f(i_P) + (k+1)\big(c(P) - m(c)\big) \right],$$

sendo o mínimo considerado, é claro, sobre o mesmo conjunto de caminhos.

Observação

Um caso particular do item 3 é

$$T_{c-m_k} T_{c-m_{k-1}} \cdots T_{c-m_1} T_{c-m_0} f(j) = \\ = \min_P \left[f(i_P) + (k+1)c(P) - \sum_{r=0}^{k} m_r \right],$$

onde o mínimo é tomado sobre todos os caminhos P em G de comprimento $k+1$ conectando vértice arbitrário i_P ao vértice fixo j.

Ademais, caso $m_k = \ldots = m_1 = m_0 = m(c)$, temos

$$T^{k+1}_{c-m(c)} f(j) = \min_P \left[f(i_P) + (k+1)\big(c(P) - m(c)\big) \right],$$

sendo o mínimo considerado, é claro, sobre o mesmo conjunto de caminhos.

Observação

Um caso particular do item 3 é

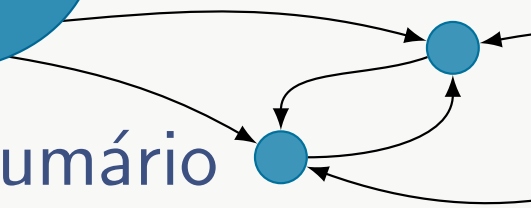
$$T_{c-m_k} T_{c-m_{k-1}} \cdots T_{c-m_1} T_{c-m_0} f(j) = \\ = \min_P \left[f(i_P) + (k+1)c(P) - \sum_{r=0}^{k} m_r \right],$$

onde o mínimo é tomado sobre todos os caminhos P em G de comprimento $k+1$ conectando vértice arbitrário i_P ao vértice fixo j.

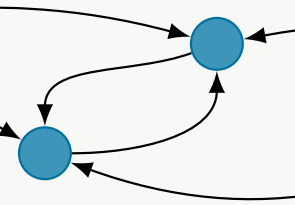
Ademais, caso $m_k = \ldots = m_1 = m_0 = m(c)$, temos

$$T^{k+1}_{c-m(c)} f(j) = \min_P \left[f(i_P) + (k+1)\big(c(P) - m(c)\big) \right],$$

sendo o mínimo considerado, é claro, sobre o mesmo conjunto de caminhos.



Sumário



Algoritmo de Floría-Griffiths

Tal algoritmo consiste em

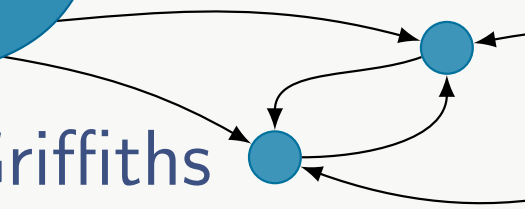
um processo iterativo que, a partir de uma função qualquer sobre os vértices do grafo e de uma cota superior para a constante cíclica, produz um corretor e determina a própria constante.

Notação

- Por uma questão de praticidade, escreveremos várias vezes $a \wedge b$ significando, como de costume, $\min\{a, b\}$;
- Adotaremos a convenção também usual em otimização de denotar, para uma função F sobre um conjunto C, por

$$\operatorname{argmin}\{F(x) : x \in C\}$$

o conjunto dos elementos $y \in C$ que verificam $F(y) = \min\{F(x) : x \in C\}$.



Algoritmo de Floría-Griffiths

Tal algoritmo consiste em

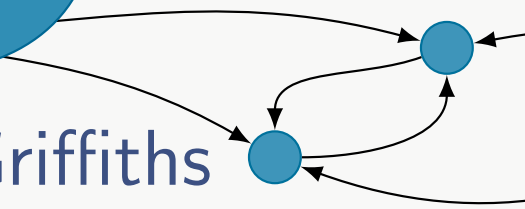
um processo iterativo que, a partir de uma função qualquer sobre os vértices do grafo e de uma cota superior para a constante cíclica, produz um corretor e determina a própria constante.

Notação

- Por uma questão de praticidade, escreveremos várias vezes $a \wedge b$ significando, como de costume, $\min\{a, b\}$;
- Adotaremos a convenção também usual em otimização de denotar, para uma função F sobre um conjunto C, por

$$\operatorname{argmin}\{F(x) : x \in C\}$$

o conjunto dos elementos $y \in C$ que verificam $F(y) = \min\{F(x) : x \in C\}$.



Algoritmo de Floría-Griffiths

Tal algoritmo consiste em

um processo iterativo que, a partir de uma função qualquer sobre os vértices do grafo e de uma cota superior para a constante cíclica, produz um corretor e determina a própria constante.

Notação

- Por uma questão de praticidade, escreveremos várias vezes $a \wedge b$ significando, como de costume, $\min\{a, b\}$;
- Adotaremos a convenção também usual em otimização de denotar, para uma função F sobre um conjunto C, por

$$\operatorname{argmin}\{F(x) : x \in C\}$$

o conjunto dos elementos $y \in C$ que verificam $F(y) = \min\{F(x) : x \in C\}$.

Algoritmo de Floría-Griffiths

Começamos fornecendo:

$$\textbf{Entradas} \begin{cases} V_0 = V(G), \\ u_0 : V(G) \to \mathbb{R} \quad \text{função arbitrária}, \\ m_0 \geq m(c). \end{cases}$$

Note que uma maneira prática de obter cota superior m_0 é tomar o mínimo de $c(P)$ com P pertencente a um subconjunto fixo de ciclos.

Na primeira iteração, obtemos:

$$\textbf{Saídas (Iterada 1)} \begin{cases} V_1 \subset V(G), \\ u_1 : V(G) \to \mathbb{R}, \quad u_1 \leq u_0, \\ m_1 \in [m(c), m_0]. \end{cases}$$

Algoritmo de Floría-Griffiths

Começamos fornecendo:

$$\textbf{Entradas} \begin{cases} V_0 = V(G), \\ u_0 : V(G) \to \mathbb{R} \quad \text{função arbitrária}, \\ m_0 \geq m(c). \end{cases}$$

Note que uma maneira prática de obter cota superior m_0 é tomar o mínimo de $c(P)$ com P pertencente a um subconjunto fixo de ciclos.

Na primeira iteração, obtemos:

$$\textbf{Saídas (Iterada 1)} \begin{cases} V_1 \subset V(G), \\ u_1 : V(G) \to \mathbb{R}, \quad u_1 \leq u_0, \\ m_1 \in [m(c), m_0]. \end{cases}$$

Algoritmo de Floría-Griffiths

Começamos fornecendo:

$$\textbf{Entradas} \begin{cases} V_0 = V(G), \\ u_0 : V(G) \to \mathbb{R} \quad \text{função arbitrária}, \\ m_0 \geq m(c). \end{cases}$$

Note que uma maneira prática de obter cota superior m_0 é tomar o mínimo de $c(P)$ com P pertencente a um subconjunto fixo de ciclos.

Na primeira iteração, obtemos:

$$\textbf{Saídas (Iterada 1)} \begin{cases} V_1 \subset V(G), \\ u_1 : V(G) \to \mathbb{R}, \quad u_1 \leq u_0, \\ m_1 \in [m(c), m_0]. \end{cases}$$

Os dois primeiros dados resultam simplesmente ao se colocar:

- $V_1 := \{j \in V(G) : T_{c-m_0} u_0(j) < u_0(j)\}$;
- $u_1(j) := \min\{u_0(j),\ T_{c-m_0} u_0(j)\}, \quad \forall\, j \in V(G)$.

Quanto à determinação da nova cota superior, devemos proceder da forma abaixo detalhada.

- Primeiro, definimos $\sigma_1 : V(G) \to V(G)$ obedecendo
  - se $j \notin V_1$, escolhemos $\sigma_1(j)$ pertencente a

$$\{i \in V(G) : M(i,j) = 1\};$$

  - se $j \in V_1$, escolhemos $\sigma_1(j)$ no conjunto

$$\operatorname{argmin}\{u_0(i) + c(i,j) - m_0 : i \text{ com } (i,j) \in A(G)\} = \operatorname{argmin}\{u_0(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Os dois primeiros dados resultam simplesmente ao se colocar:

- $V_1 := \{j \in V(G) : T_{c-m_0} u_0(j) < u_0(j)\}$;
- $u_1(j) := \min\{u_0(j),\ T_{c-m_0} u_0(j)\}, \quad \forall j \in V(G)$.

Quanto à determinação da nova cota superior, devemos proceder da forma abaixo detalhada.

- Primeiro, definimos $\sigma_1 : V(G) \to V(G)$ obedecendo
  - se $j \notin V_1$, escolhemos $\sigma_1(j)$ pertencente a

$$\{i \in V(G) : M(i,j) = 1\};$$

  - se $j \in V_1$, escolhemos $\sigma_1(j)$ no conjunto

$$\operatorname{argmin}\{u_0(i) + c(i,j) - m_0 : i \text{ com } (i,j) \in A(G)\} = \operatorname{argmin}\{u_0(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Os dois primeiros dados resultam simplesmente ao se colocar:

- $V_1 := \{j \in V(G) : T_{c-m_0} u_0(j) < u_0(j)\}$;
- $u_1(j) := \min\{u_0(j),\ T_{c-m_0} u_0(j)\}, \quad \forall j \in V(G)$.

Quanto à determinação da nova cota superior, devemos proceder da forma abaixo detalhada.

- Primeiro, definimos $\sigma_1 : V(G) \to V(G)$ obedecendo
  - se $j \notin V_1$, escolhemos $\sigma_1(j)$ pertencente a

$$\{i \in V(G) : M(i,j) = 1\};$$

  - se $j \in V_1$, escolhemos $\sigma_1(j)$ no conjunto

$$\operatorname{argmin}\{u_0(i) + c(i,j) - m_0 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_0(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Os dois primeiros dados resultam simplesmente ao se colocar:

- $V_1 := \{j \in V(G) : T_{c-m_0} u_0(j) < u_0(j)\}$;
- $u_1(j) := \min\{u_0(j),\ T_{c-m_0} u_0(j)\}, \quad \forall j \in V(G)$.

Quanto à determinação da nova cota superior, devemos proceder da forma abaixo detalhada.

- Primeiro, definimos $\sigma_1 : V(G) \to V(G)$ obedecendo
  - se $j \notin V_1$, escolhemos $\sigma_1(j)$ pertencente a
$$\{i \in V(G) : M(i,j) = 1\};$$
  - se $j \in V_1$, escolhemos $\sigma_1(j)$ no conjunto
$$\begin{aligned}\operatorname{argmin}\{u_0(i) + c(i,j) - m_0 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_0(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.\end{aligned}$$

Para cada $j \in V_1$, consideramos $k_1(j) := j$ e $k_2(j) := \sigma_1(k_1(j))$.
Se $j = k_1(j) = k_2(j)$, temos um ciclo P_1^j,

$$j = k_2(j) \xrightarrow{G} k_1(j) = j.$$

Colocamos então $m_1 := m_0 \wedge \min\{c(P_1^j) : j \in V_1\}$.

Na segunda iteração, encontramos

$$\textbf{Saídas (Iterada 2)} \quad \begin{cases} V_2 \subset V(G), \\ u_2 : V(G) \to \mathbb{R}, \quad u_2 \leq u_1, \\ m_2 \in [m(c), m_1]. \end{cases}$$

Construímos também

- $V_2 := \{j \in V(G) : T_{c-m_1} u_1(j) < u_1(j)\}$;

Para cada $j \in V_1$, consideramos $k_1(j) := j$ e $k_2(j) := \sigma_1(k_1(j))$.
Se $j = k_1(j) = k_2(j)$, temos um ciclo P_1^j,

$$j = k_2(j) \xrightarrow{G} k_1(j) = j.$$

Colocamos então $m_1 := m_0 \wedge \min\{c(P_1^j) : j \in V_1\}$.

Na segunda iteração, encontramos

Saídas (Iterada 2) $\begin{cases} V_2 \subset V(G), \\ u_2 : V(G) \to \mathbb{R}, \quad u_2 \leq u_1, \\ m_2 \in [m(c), m_1]. \end{cases}$

Construção dos V_2 e u_2:

- $V_2 := \{j \in V(G) : T_{c-m_1} u_1(j) < u_1(j)\}$;

Para cada $j \in V_1$, consideramos $k_1(j) := j$ e $k_2(j) := \sigma_1(k_1(j))$.
Se $j = k_1(j) = k_2(j)$, temos um ciclo P_1^j,

$$j = k_2(j) \xrightarrow{G} k_1(j) = j.$$

Colocamos então $m_1 := m_0 \wedge \min\{c(P_1^j) : j \in V_1\}$.

Na segunda iteração, encontramos

Saídas (Iterada 2) $\begin{cases} V_2 \subset V(G), \\ u_2 : V(G) \to \mathbb{R}, \quad u_2 \leq u_1, \\ m_2 \in [m(c), m_1]. \end{cases}$

Como antes, fazemos:

- $V_2 := \{j \in V(G) : T_{c-m_1} u_1(j) < u_1(j)\}$;
- $u_2(j) := \min\{u_1(j),\ T_{c-m_1} u_1(j)\}, \quad \forall j \in V(G)$.

Para cada $j \in V_1$, consideramos $k_1(j) := j$ e $k_2(j) := \sigma_1(k_1(j))$.
Se $j = k_1(j) = k_2(j)$, temos um ciclo P_1^j,

$$j = k_2(j) \xrightarrow{G} k_1(j) = j.$$

Colocamos então $m_1 := m_0 \wedge \min\{c(P_1^j) : j \in V_1\}$.

Na segunda iteração, encontramos

Saídas (Iterada 2) $\begin{cases} V_2 \subset V(G), \\ u_2 : V(G) \to \mathbb{R}, \quad u_2 \leq u_1, \\ m_2 \in [m(c), m_1]. \end{cases}$

Como antes, fazemos:

- $V_2 := \{j \in V(G) : T_{c-m_1} u_1(j) < u_1(j)\}$;
- $u_2(j) := \min\{u_1(j),\ T_{c-m_1} u_1(j)\}, \quad \forall j \in V(G)$.

Para cada $j \in V_1$, consideramos $k_1(j) := j$ e $k_2(j) := \sigma_1(k_1(j))$.
Se $j = k_1(j) = k_2(j)$, temos um ciclo P_1^j,

$$j = k_2(j) \xrightarrow{G} k_1(j) = j.$$

Colocamos então $m_1 := m_0 \wedge \min\{c(P_1^j) : j \in V_1\}$.

Na segunda iteração, encontramos

$$\textbf{Saídas (Iterada 2)} \quad \begin{cases} V_2 \subset V(G), \\ u_2 : V(G) \to \mathbb{R}, \quad u_2 \leq u_1, \\ m_2 \in [m(c), m_1]. \end{cases}$$

Como antes, fazemos:

- $V_2 := \{j \in V(G) : T_{c-m_1} u_1(j) < u_1(j)\}$;
- $u_2(j) := \min\{u_1(j),\ T_{c-m_1} u_1(j)\}, \quad \forall j \in V(G)$.

Já com respeito à cota superior, procedemos como segue.

- Definimos $\sigma_2 : V(G) \to V(G)$ satisfazendo
  - se $j \notin V_2$, colocamos $\sigma_2(j) = \sigma_1(j)$;
  - se $j \in V_2$, escolhemos $\sigma_2(j)$ pertencente a

$$\operatorname{argmin}\{u_1(i) + c(i,j) - m_1 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_1(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Para cada $j \in V_2$, definimos

$$k_1(j) := j, \quad k_2(j) := \sigma_2(k_1(j)) \quad \text{e} \quad k_3(j) := \sigma_2(k_2(j)).$$

Se $j = k_1(j) = k_2(j)$ ou $j = k_1(j) = k_3(j)$, temos um ciclo P_2^j,

$$k_2(j) \xrightarrow{G} k_1(j) \quad \text{ou} \quad k_3(j) \xrightarrow{G} k_2(j) \xrightarrow{G} k_1(j).$$

Introduzimos finalmente $m_2 := m_1 \wedge \min\{c(P_2^j) : j \in V_2\}$.

Já com respeito à cota superior, procedemos como segue.

- Definimos $\sigma_2 : V(G) \to V(G)$ satisfazendo
  - se $j \notin V_2$, colocamos $\sigma_2(j) = \sigma_1(j)$;
  - se $j \in V_2$, escolhemos $\sigma_2(j)$ pertencente a

$$\operatorname{argmin}\{u_1(i) + c(i,j) - m_1 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_1(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Para cada $j \in V_2$, definimos

$$k_1(j) := j, \quad k_2(j) := \sigma_2(k_1(j)) \quad \text{e} \quad k_3(j) := \sigma_2(k_2(j)).$$

Se $j = k_1(j) = k_2(j)$ ou $j = k_1(j) = k_3(j)$, temos um ciclo P_2^j,

$$k_2(j) \xrightarrow{G} k_1(j) \quad \text{ou} \quad k_3(j) \xrightarrow{G} k_2(j) \xrightarrow{G} k_1(j).$$

Introduzimos finalmente $m_2 := m_1 \wedge \min\{c(P_2^j) : j \in V_2\}$.

Já com respeito à cota superior, procedemos como segue.

- Definimos $\sigma_2 : V(G) \to V(G)$ satisfazendo
  - se $j \notin V_2$, colocamos $\sigma_2(j) = \sigma_1(j)$;
  - se $j \in V_2$, escolhemos $\sigma_2(j)$ pertencente a

$$\begin{aligned}\operatorname{argmin}\{u_1(i) + c(i,j) - m_1 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_1(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.\end{aligned}$$

Para cada $j \in V_2$, definimos

$$k_1(j) := j, \quad k_2(j) := \sigma_2(k_1(j)) \quad \text{e} \quad k_3(j) := \sigma_2(k_2(j)).$$

Se $j = k_1(j) = k_2(j)$ ou $j = k_1(j) = k_3(j)$, temos um ciclo P_2^j,

$$k_2(j) \xrightarrow{G} k_1(j) \quad \text{ou} \quad k_3(j) \xrightarrow{G} k_2(j) \xrightarrow{G} k_1(j).$$

Introduzimos finalmente $m_2 := m_1 \wedge \min\{c(P_2^j) : j \in V_2\}$.

Já com respeito à cota superior, procedemos como segue.

- Definimos $\sigma_2 : V(G) \to V(G)$ satisfazendo
  - se $j \notin V_2$, colocamos $\sigma_2(j) = \sigma_1(j)$;
  - se $j \in V_2$, escolhemos $\sigma_2(j)$ pertencente a

$$\begin{aligned}\operatorname{argmin}\{u_1(i) + c(i,j) - m_1 : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_1(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.\end{aligned}$$

Para cada $j \in V_2$, definimos

$$k_1(j) := j, \quad k_2(j) := \sigma_2(k_1(j)) \quad \text{e} \quad k_3(j) := \sigma_2(k_2(j)).$$

Se $j = k_1(j) = k_2(j)$ ou $j = k_1(j) = k_3(j)$, temos um ciclo P_2^j,

$$k_2(j) \xrightarrow{G} k_1(j) \quad \text{ou} \quad k_3(j) \xrightarrow{G} k_2(j) \xrightarrow{G} k_1(j).$$

Introduzimos finalmente $m_2 := m_1 \wedge \min\{c(P_2^j) : j \in V_2\}$.

Após r iterações, o algoritmo produz:

Saídas (Iterada r) $\begin{cases} V_r \subset V(G), \\ u_r : V(G) \to \mathbb{R}, \quad u_r \leq u_{r-1} \leq \ldots \leq u_0, \\ m_r \in [m(c), m_{r-1}]. \end{cases}$

Colocamos:

- $V_r := \{j \in V(G) : T_{c-m_{r-1}} u_{r-1}(j) < u_{r-1}(j)\}$;
- $u_r(j) := \min\{u_{r-1}(j),\ T_{c-m_{r-1}} u_{r-1}(j)\}, \quad \forall j \in V(G)$.

- Definimos $\sigma_r : V(G) \to V(G)$ verificando
  - se $j \notin V_r$, fazemos $\sigma_r(j) = \sigma_{r-1}(j)$;

Após r iterações, o algoritmo produz:

$$\textbf{Saídas (Iterada r)} \quad \begin{cases} V_r \subset V(G), \\ u_r : V(G) \to \mathbb{R}, \quad u_r \leq u_{r-1} \leq \ldots \leq u_0, \\ m_r \in [m(c), m_{r-1}]. \end{cases}$$

Colocamos:

- $V_r := \{j \in V(G) : T_{c-m_{r-1}} u_{r-1}(j) < u_{r-1}(j)\}$;
- $u_r(j) := \min\{u_{r-1}(j),\ T_{c-m_{r-1}} u_{r-1}(j)\}, \quad \forall\, j \in V(G).$

Além disso, a definição da nova cota superior ocorre de acordo com os passos gerais descritos abaixo.

- Definimos $\sigma_r : V(G) \to V(G)$ verificando
  - se $j \notin V_r$, fazemos $\sigma_r(j) = \sigma_{r-1}(j)$;
  - se $j \in V_r$, escolhemos $\sigma_r(j)$ no conjunto

$$\operatorname{argmin}\{u_{r-1}(i) + c(i,j) - m_{r-1} : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_{r-1}(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Após r iterações, o algoritmo produz:

$$\textbf{Saídas (Iterada r)} \quad \begin{cases} V_r \subset V(G), \\ u_r : V(G) \to \mathbb{R}, \quad u_r \leq u_{r-1} \leq \ldots \leq u_0, \\ m_r \in [m(c), m_{r-1}]. \end{cases}$$

Colocamos:

- $V_r := \{j \in V(G) : T_{c-m_{r-1}} u_{r-1}(j) < u_{r-1}(j)\}$;
- $u_r(j) := \min\{u_{r-1}(j),\ T_{c-m_{r-1}} u_{r-1}(j)\}, \quad \forall\, j \in V(G).$

Além disso, a definição da nova cota superior ocorre de acordo com os passos gerais descritos abaixo.

- Definimos $\sigma_r : V(G) \to V(G)$ verificando
  - se $j \notin V_r$, fazemos $\sigma_r(j) = \sigma_{r-1}(j)$;
  - se $j \in V_r$, escolhemos $\sigma_r(j)$ no conjunto

$$\operatorname{argmin}\{u_{r-1}(i) + c(i,j) - m_{r-1} : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_{r-1}(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Após r iterações, o algoritmo produz:

$$\textbf{Saídas (Iterada r)} \quad \begin{cases} V_r \subset V(G), \\ u_r : V(G) \to \mathbb{R}, \quad u_r \leq u_{r-1} \leq \ldots \leq u_0, \\ m_r \in [m(c), m_{r-1}]. \end{cases}$$

Colocamos:

- $V_r := \{j \in V(G) : T_{c-m_{r-1}} u_{r-1}(j) < u_{r-1}(j)\}$;
- $u_r(j) := \min\{u_{r-1}(j),\ T_{c-m_{r-1}} u_{r-1}(j)\}, \quad \forall\, j \in V(G).$

Além disso, a definição da nova cota superior ocorre de acordo com os passos gerais descritos abaixo.

- Definimos $\sigma_r : V(G) \to V(G)$ verificando
  - se $j \notin V_r$, fazemos $\sigma_r(j) = \sigma_{r-1}(j)$;
  - se $j \in V_r$, escolhemos $\sigma_r(j)$ no conjunto

$$\operatorname{argmin}\{u_{r-1}(i) + c(i,j) - m_{r-1} : i \text{ com } (i,j) \in A(G)\} = \\ = \operatorname{argmin}\{u_{r-1}(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}.$$

Após r iterações, o algoritmo produz:

$$\textbf{Saídas (Iterada r)} \quad \begin{cases} V_r \subset V(G), \\ u_r : V(G) \to \mathbb{R}, \quad u_r \leq u_{r-1} \leq \ldots \leq u_0, \\ m_r \in [m(c), m_{r-1}]. \end{cases}$$

Colocamos:

- $V_r := \{j \in V(G) : T_{c-m_{r-1}} u_{r-1}(j) < u_{r-1}(j)\}$;
- $u_r(j) := \min\{u_{r-1}(j),\ T_{c-m_{r-1}} u_{r-1}(j)\}, \quad \forall j \in V(G)$.

Além disso, a definição da nova cota superior ocorre de acordo com os passos gerais descritos abaixo.

- Definimos $\sigma_r : V(G) \to V(G)$ verificando
  - se $j \notin V_r$, fazemos $\sigma_r(j) = \sigma_{r-1}(j)$;
  - se $j \in V_r$, escolhemos $\sigma_r(j)$ no conjunto

$$\begin{aligned} &\operatorname{argmin}\{u_{r-1}(i) + c(i,j) - m_{r-1} : i \text{ com } (i,j) \in A(G)\} = \\ &\qquad = \operatorname{argmin}\{u_{r-1}(i) + c(i,j) : i \text{ com } (i,j) \in A(G)\}. \end{aligned}$$

Para cada $j \in V_r$, consideramos

$$k_1(j) := j, \quad k_2(j) := \sigma_r(k_1(j)), \quad \ldots, \quad k_{r+1}(j) := \sigma_r(k_r(j)).$$

Se $k_1(j) = k_s(j)$ para algum $s \in \{2, \ldots, r+1\}$, temos então um ciclo P_r^j,

$$k_s(j) \xrightarrow{G} k_{s-1}(j) \xrightarrow{G} \ldots \xrightarrow{G} k_1(j).$$

Colocamos então $m_r := m_{r-1} \wedge \min\{c(P_r^j) : j \in V_r\}$.

Por último, temos:

A condição de finalização do algoritmo é dada pela exigência:

Parada $\left\{ V_{r+1} = \emptyset. \right.$

Para cada $j \in V_r$, consideramos

$$k_1(j) := j, \quad k_2(j) := \sigma_r(k_1(j)), \quad \ldots, \quad k_{r+1}(j) := \sigma_r(k_r(j)).$$

Se $k_1(j) = k_s(j)$ para algum $s \in \{2, \ldots, r+1\}$, temos então um ciclo P_r^j,

$$k_s(j) \xrightarrow{G} k_{s-1}(j) \xrightarrow{G} \ldots \xrightarrow{G} k_1(j).$$

Colocamos então $m_r := m_{r-1} \wedge \min\{c(P_r^j) : j \in V_r\}$.

Por último, temos:

A condição de finalização do algoritmo é dada pela exigência:

Parada $\left\{ V_{r+1} = \emptyset. \right.$

Para cada $j \in V_r$, consideramos

$$k_1(j) := j, \quad k_2(j) := \sigma_r(k_1(j)), \quad \ldots, \quad k_{r+1}(j) := \sigma_r(k_r(j)).$$

Se $k_1(j) = k_s(j)$ para algum $s \in \{2, \ldots, r+1\}$, temos então um ciclo P_r^j,

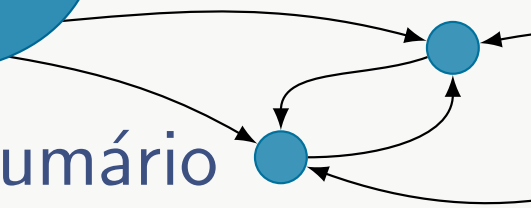
$$k_s(j) \xrightarrow{G} k_{s-1}(j) \xrightarrow{G} \ldots \xrightarrow{G} k_1(j).$$

Colocamos então $m_r := m_{r-1} \wedge \min\{c(P_r^j) : j \in V_r\}$.

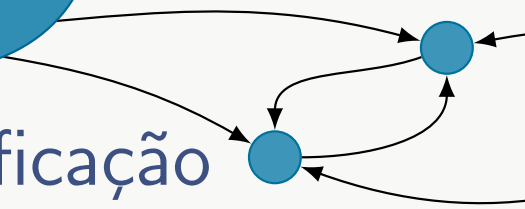
Por último, temos:

A condição de finalização do algoritmo é dada pela exigência:

Parada $\Big\{ V_{r+1} = \emptyset.$



Sumário



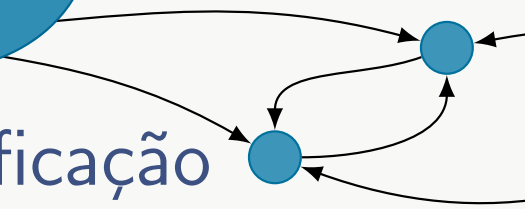
Comentários sobre a verificação

Supondo que o algoritmo funcione, a pergunta básica que devemos imediatamente responder é

se (m_r, u_r) é um par (constante cíclica minimal, corretor).

Ora, note que

$$V_{r+1} = \emptyset \quad \Leftrightarrow \quad u_r \leq T_{c-m_r} u_r .$$



Comentários sobre a verificação

Supondo que o algoritmo funcione, a pergunta básica que devemos imediatamente responder é

se (m_r, u_r) é um par (constante cíclica minimal, corretor).

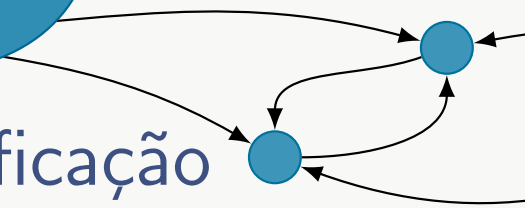
Ora, note que

$$V_{r+1} = \emptyset \quad \Leftrightarrow \quad u_r \leq T_{c-m_r} u_r.$$

Para um ciclo $P : i_0 \xrightarrow{G} i_1 \xrightarrow{G} i_2 \xrightarrow{G} \ldots \xrightarrow{G} i_k = i_0$ (com $k \geq 1$) em G, as desigualdades

$$\begin{array}{rcl} u_r(i_k) & \leq & u_r(i_{k-1}) + c(i_{k-1}, i_k) - m_r \\ u_r(i_{k-1}) & \leq & u_r(i_{k-2}) + c(i_{k-2}, i_{k-1}) - m_r \\ & \vdots & \\ u_r(i_1) & \leq & u_r(i_0) + c(i_0, i_1) - m_r \end{array}$$

implicam $m_r \leq c(P)$.



Comentários sobre a verificação

Supondo que o algoritmo funcione, a pergunta básica que devemos imediatamente responder é

se (m_r, u_r) é um par (constante cíclica minimal, corretor).

Ora, note que

$$V_{r+1} = \emptyset \quad \Leftrightarrow \quad u_r \leq T_{c-m_r} u_r.$$

Para um ciclo $P : i_0 \xrightarrow{G} i_1 \xrightarrow{G} i_2 \xrightarrow{G} \ldots \xrightarrow{G} i_k = i_0$ (com $k \geq 1$) em G, as desigualdades

$$\begin{array}{rcl} u_r(i_k) & \leq & u_r(i_{k-1}) + c(i_{k-1}, i_k) - m_r \\ u_r(i_{k-1}) & \leq & u_r(i_{k-2}) + c(i_{k-2}, i_{k-1}) - m_r \\ & \vdots & \\ u_r(i_1) & \leq & u_r(i_0) + c(i_0, i_1) - m_r \end{array}$$

implicam $m_r \leq c(P)$.

Uma vez que P é um ciclo arbitrário, temos que

$$m_r \leq m(c).$$

Mas por construção $m_r \geq m(c)$. Logo, por antissimetria,

$$u_r \leq T_{c-m(c)} u_r$$

Suponha que u_r é finalmente periódico.

Um fato a ter em mente é que,

quando um ciclo que minimiza o custo está entre os dados pela k-ésima etapa do algoritmo, então são necessários no máximo $|V|$ passos adicionais para que o algoritmo pare.

Uma vez que P é um ciclo arbitrário, temos que

$$m_r \leq m(c).$$

Mas por construção $m_r \geq m(c)$. Logo, $m_r = m(c)$ e a desigualdade

$$u_r \leq T_{c-m(c)} u_r$$

significa que u_r é um corretor.

Uma das propriedades importantes é que

quando um ciclo que minimiza o custo está entre os ciclos P_s^j de alguma etapa s do algoritmo, então são necessárias no máximo $\#V(G)$ iterações adicionais para que o algoritmo pare.

Uma vez que P é um ciclo arbitrário, temos que

$$m_r \leq m(c).$$

Mas por construção $m_r \geq m(c)$. Logo, $m_r = m(c)$ e a desigualdade

$$u_r \leq T_{c-m(c)} u_r$$

significa que u_r é um corretor.

Um fato a ter em mente é que,

quando um ciclo que minimiza o custo está entre os ciclos P_s^j de alguma etapa s do algoritmo, então são necessárias no máximo $\#V(G)$ iterações adicionais para que o algoritmo pare.

Uma vez que P é um ciclo arbitrário, temos que

$$m_r \leq m(c).$$

Mas por construção $m_r \geq m(c)$. Logo, $m_r = m(c)$ e a desigualdade

$$u_r \leq T_{c-m(c)} u_r$$

significa que u_r é um corretor.

Um fato a ter em mente é que,

quando um ciclo que minimiza o custo está entre os ciclos P_s^j de alguma etapa s do algoritmo, então são necessárias no máximo $\#V(G)$ iterações adicionais para que o algoritmo pare.

Isto está mais precisamente apresentado na próxima proposição.

Proposição

Denote $n = \#V(G)$. Se ocorrer $m_s = m(c)$, então $V_{r+1} = \emptyset$ para algum $r \in \{s, s+1, \ldots, s+n\}$.

A validação do algoritmo fica, portanto, condicionada à presença de algum ciclo que minimiza o custo entre os ciclos P_s^j de alguma etapa s.

Observação

O leitor interessado em argumentos completos para assegurar que a condição de finalização do algoritmo de Floría-Griffiths é alcançada pode consultar o artigo original [1].

Isto está mais precisamente apresentado na próxima proposição.

Proposição

Denote $n = \#V(G)$. Se ocorrer $m_s = m(c)$, então $V_{r+1} = \emptyset$ para algum $r \in \{s, s+1, \ldots, s+n\}$.

A validação do algoritmo fica, portanto, condicionada à presença de algum ciclo que minimiza o custo entre os ciclos P_s^j de alguma etapa s.

Observação

O leitor interessado em argumentos completos para assegurar que a condição de finalização do algoritmo de Floría-Griffiths é alcançada pode consultar o artigo original [1].

Isto está mais precisamente apresentado na próxima proposição.

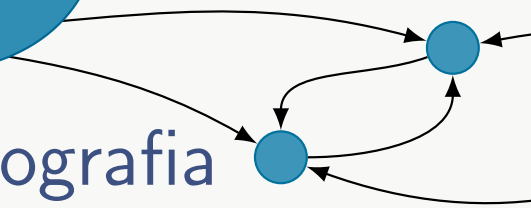
Proposição

Denote $n = \#V(G)$. Se ocorrer $m_s = m(c)$, então $V_{r+1} = \emptyset$ para algum $r \in \{s, s+1, \ldots, s+n\}$.

A validação do algoritmo fica, portanto, condicionada à presença de algum ciclo que minimiza o custo entre os ciclos P_s^j de alguma etapa s.

Observação

O leitor interessado em argumentos completos para assegurar que a condição de finalização do algoritmo de Floría-Griffiths é alcançada pode consultar o artigo original [1].

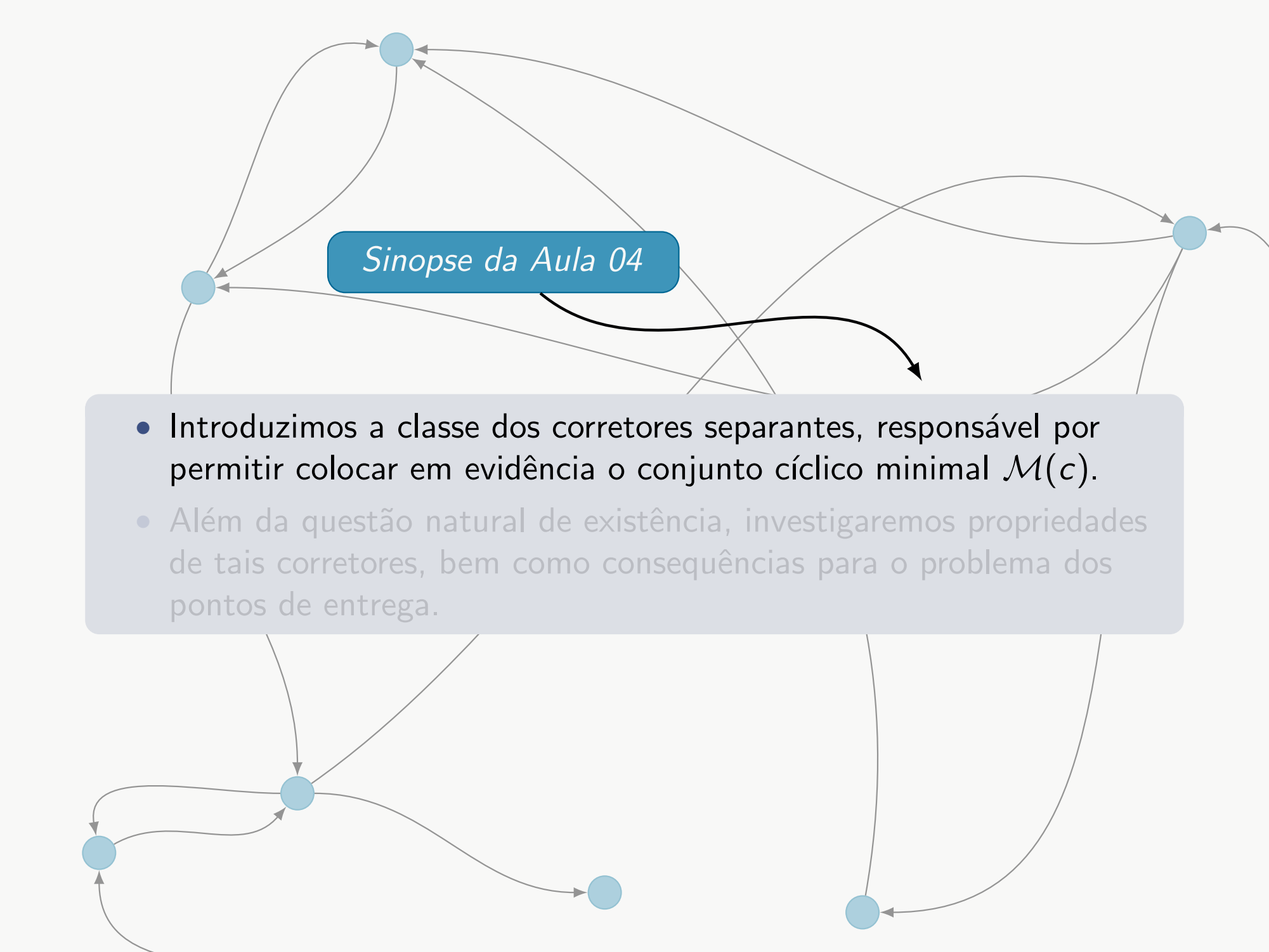


Bibliografia

E. Garibaldi e J. T. A. Gomes,
Otimização de Médias sobre Grafos Orientados,
Coleção publicações matemáticas (29 CBM) **12**, IMPA, 2013.

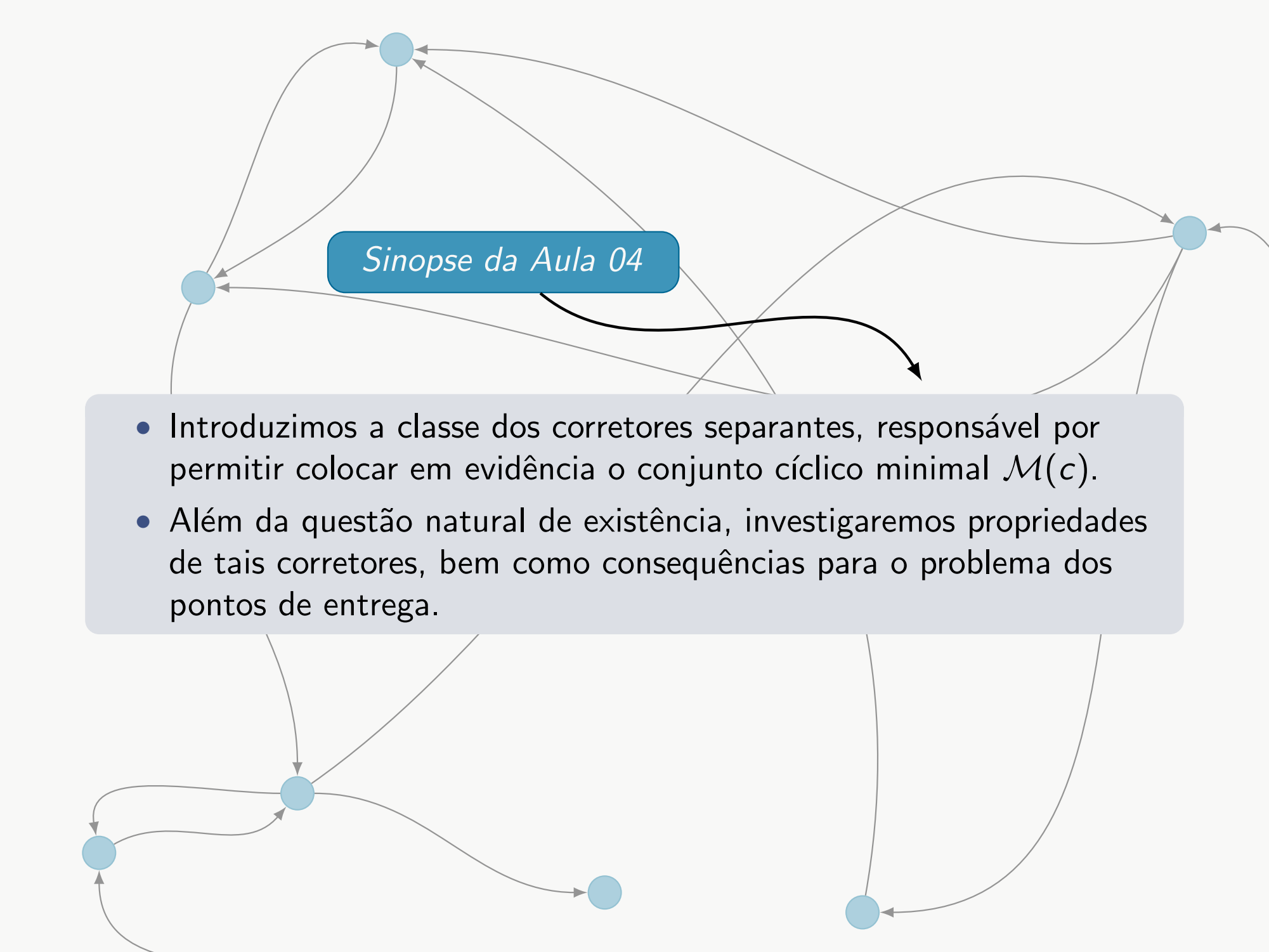
Seções: 3.1 e 3.2 (páginas 37 a 44);

L. M. Floría e R. B. Griffiths,
Numerical procedure for solving a minimization eigenvalue problem,
Numerische Mathematik **55** (1989), 565-574.



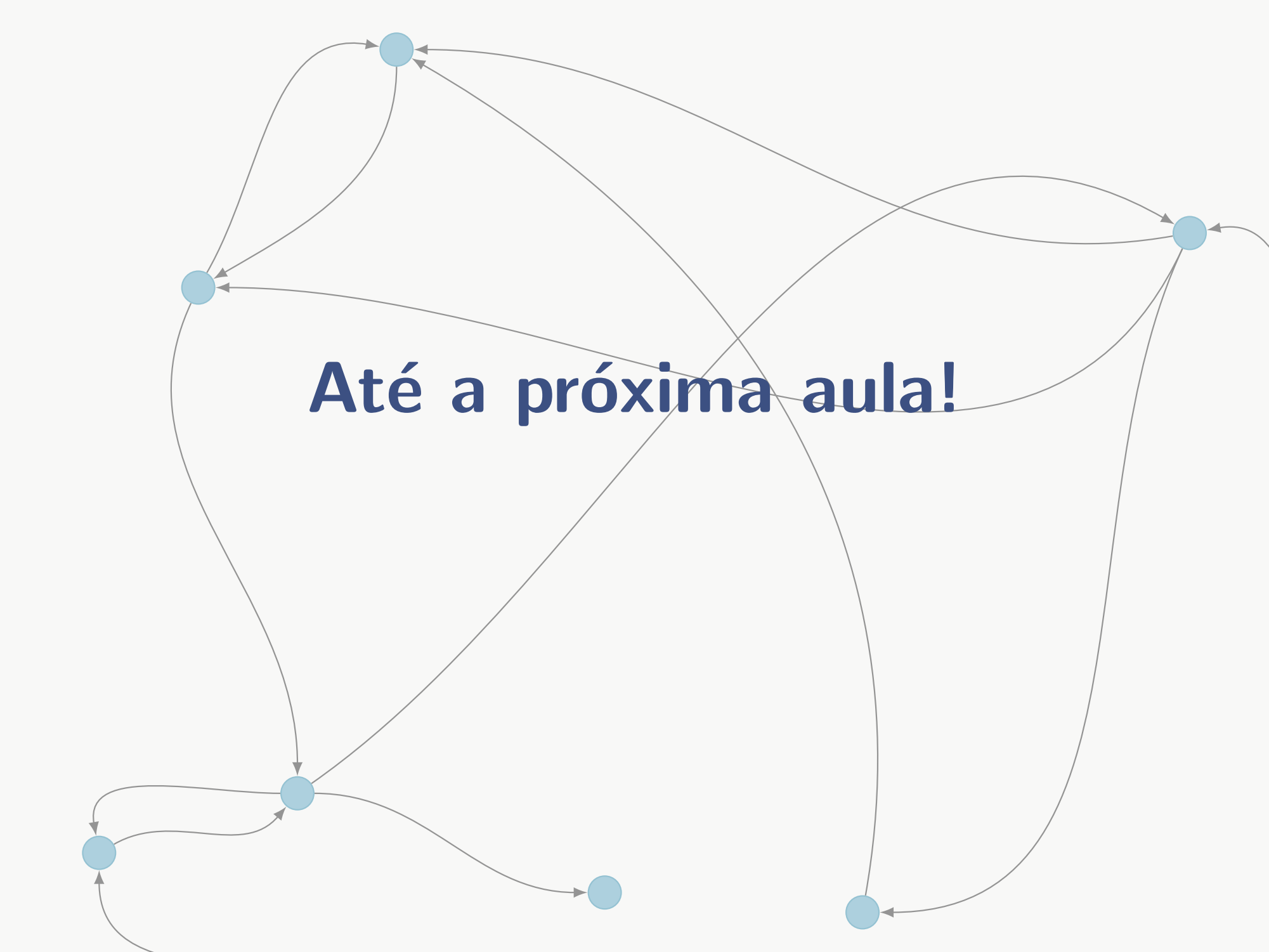
Sinopse da Aula 04

- Introduzimos a classe dos corretores separantes, responsável por permitir colocar em evidência o conjunto cíclico minimal $\mathcal{M}(c)$.
- Além da questão natural de existência, investigaremos propriedades de tais corretores, bem como consequências para o problema dos pontos de entrega.



Sinopse da Aula 04

- Introduzimos a classe dos corretores separantes, responsável por permitir colocar em evidência o conjunto cíclico minimal $\mathcal{M}(c)$.
- Além da questão natural de existência, investigaremos propriedades de tais corretores, bem como consequências para o problema dos pontos de entrega.


Até a próxima aula!